TILING SURFACES
WITH STRAIGHT STRIPS

by

E. Joseph Kahlert

B.A.Sc, University of British Columbia, 1992

M.B.A., Nova Southeastern University, 1998

A Thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science
in the School
of
Computing Science

SIMON FRASER UNIVERSITY
Summer 2009

APPROVAL

Name: E. Joseph Kahlert

Degree: Master of Science

Title of Thesis: Tiling Surfaces With Straight Strips

Examining Committee: Dr. Arthur Kirkpatrick
Chair

Dr. Richard (Hao) Zhang, Senior Supervisor

Dr. Binay Bhattacharya, Supervisor

Dr. Shahram Payandeh, SFU Examiner

Date Approved: May 20, 2009

Declaration of
Partial Copyright Licence

Abstract

We present an algorithm computing geodesic curves partitioning an open mesh into segments which can be approximated using long, trimmed strips of material possessing a prescribed width. We call this *straight strip tiling* of a curved surface, with applications such as the surfacing of curved roofs. Our strips are straight since they conform to being rectangular, in contrast to possibly highly curved strip segments studied for developable surface decomposition. Starting from a geodesic curve defined by a user-specified starting point and direction we compute recursively neighbouring geodesics which respect the constraints and lead to optimal material usage. Our algorithm is exact with respect to the polyhedral geometry of the mesh and runs on a variety of surfaces with modest time complexity of $O(n^{1.5})$, where n is the mesh size. We extend the algorithm by relaxing the constraint that geodesics span the mesh allowing application to meshes with greater undulation.

Keywords: computational geometry; geometric shape modeling; geodesic curves; straight strip tiling; panelling; roofing

To my wife Tonya and my kids Alexis, John Paul, Miranda and Aurora.

"You can always become better."

— Tiger Woods

Acknowledgments

I would like to thank Dr. Richard (Hao) Zhang for navigating me through the uncharted and often tumultuous waters of this thesis project. Without his firm and guiding hand I surely would have given up hope long ago. I would like to thank John Gudaitis of Automated Systems Research because without his moral and financial support my studies and this research would not have been possible. I would like to thank the members of my examining committee, Binay Bhattacharya and Shahram Payandeh, for their time spent reviewing and providing feedback on my work. I would also like to thank my fellow students at the GRaphics Usability and VIsualization lab of the School of Computing Science — especially Ramsay Dyer and Matt Olson — not only for their support and companionship but also their contributions to this work. Finally I would like to thank my wife Tonya for her love and support without whom I would be lost.

Contents

List of Tables

List of Figures

Preface

Shortly after I began working with Automated Systems Research — a supplier of estimating software for roofing contractors — in January 2002, I estimated for ASR's principal, John Gudaitis, "two weeks - possibly three" to re-design and re-implement the roof plane generation algorithm for their sloped roof estimation product TopView. Six years later, with the help of my supervisor Dr. Richard Zhang, fellow student Matt Olson and existing work on straight skeletons I was finally able to solve this deceptively challenging problem once and for all. Thankfully, John did not hold me to my original estimate.

With this the only significant remaining geometrical problem on planar roofs now solved, Richard and I were forced to turn our attention to curved roofs for research worthy of a master's thesis. Luckily John directed us to a photograph of the Southern Cross Station (Figure 1.2) as the prime example of a complex curved roof. After showing it to my fellow classmates in the GRaphics Usability and VIsualization lab of the School of Computing Science it became clear that the roof panels on its surface followed an interesting path and a thesis was born...

Chapter 1

Introduction

Anyone who has made a paper maché model knows that a curved surface can be formed using strips of flexible material. If these paper strips are trimmed to eliminate overlap, the result is a *strip tiling*. In industrial settings, whether it be for building shells, ship hulls, timber construction or wig fabrication, the construction material is typically available in long rectangular strips or spools. Such strips can be cut, trimmed and joined together to approximate a surface.

In this paper, we are interested in the following *straight strip tiling* problem: given a curved surface patch without holes, approximate it using an adjoining set of *straight* strips satisfying a width constraint. By straight we mean that each strip must fit within a long rectangular piece of material with a given width δ; see Figure 1.1. In other words, the tiling strip can be obtained from the rectangular piece by cutting and trimming. Its close conformation to the rectangle makes it "straight". Note here that there is no constraint on the length of the rectangles. In a practical application such as roof building, we can imagine the above scenario for approximating a roof like the one in Figure 1.2. Needless to say, reducing material waste is of paramount importance.

1.1 Overview of our approach

A naive solution to this problem would be polygonal meshing, tiling the input surface using planar facets each of which is sufficiently small to fit within the material width. However, this does not exploit the long length of the material and thus leads to more cutting and joining than is necessary.

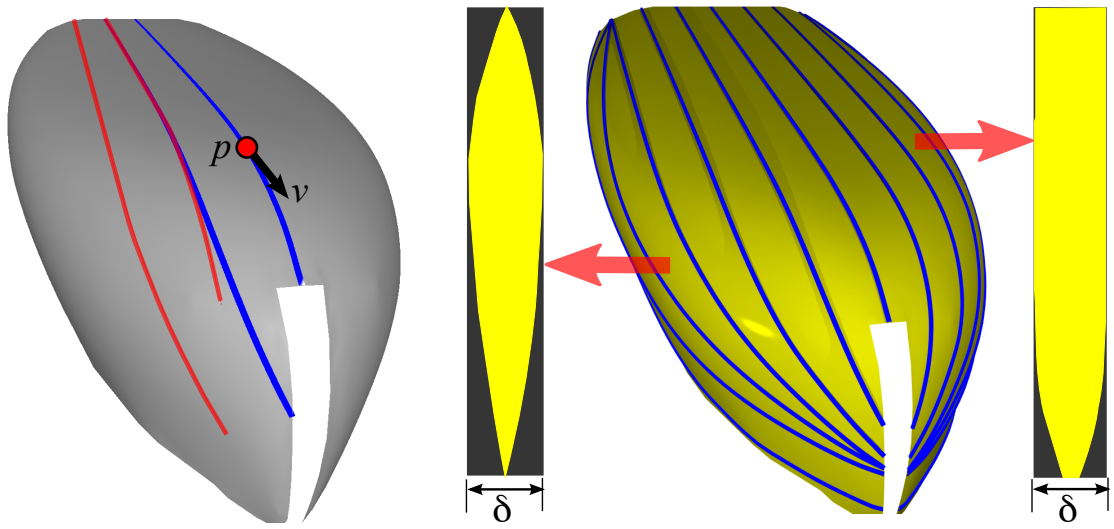


Figure 1.1: Straight strip tiling of a ship hull surface. (a) User input is given as a point p (red dot) and an orientation v, resulting in the initial geodesic passing through p (blue curve). An intermediate processing step showing the geodesic found in the first iteration (second blue curve) along with the upper bound curves (in red) being used to search for the next geodesics is shown as well. (b) The complete tiling and two resulting strip widths plotted over-top of rectangular material pieces showing that the δ width constraint is satisfied.

Planar mesh parameterization is also an option, whereby the given surface is first flattened so as to minimize distortion and then straight lines in the parameter domain define the strip boundaries which are mapped back onto the original surface. While excellent algorithms for mesh parameterization exist [22], we will see in Section 2.4 that they do not provide satisfactory solutions to respect the straightness criterion.

Consider for example the case of a hemi-spherical mesh. Conventional mesh parameterization schemes would flatten the input surface into a planar circular patch while distributing distortion uniformly across. The resulting strips on the hemisphere, obtained using curves corresponding to straight line boundaries in the parameter domain, when approximately developed, will generally be "curved" in that they will not fit well within the long material rectangles(Figure 2.1).

Recent work of Pottmann et al. [20] takes as input a set of curves on a surface to decompose it into planar strips which together provide a good surface approximation. If one of these curve families consist of *geodesics*, then this algorithm can be adapted to produce

strips with "approximately straight development".

Making use of this work, we frame the problem we wish to solve as follows: on a given mesh patch, find a set of non-intersecting geodesic curves separated by no more than the prescribed material width δ (measured geodesically) from each other and from the surface boundary so that the resulting strips, when developed, span the width of the bounded-width rectangles as much as possible (to minimize material waste).

Finding a globally optimal solution to such a problem is beyond the scope of this thesis. Instead, we solve a greedy version which must be initialized with a point and orientation on the surface. Given this input provided by the user, we can launch an initial source geodesic curve from which we can recursively find neighbouring geodesics with the properties mentioned until the surface is covered with geodesic strips of bounded width; refer to Figure 1.1.

1.2 Motivation

It is well known that when a narrow strip of flexible (but un-stretchable) material is placed in contact with a smooth surface, it will follow a geodesic path along that surface [19]. To understand why this is true, consider that while geodesics are more commonly known as shortest paths on surfaces (which is true, but only locally), their *straightness* property — i.e. the fact that they curve only in the direction of the surface normal — always holds for smooth surfaces. In other words, from the point of view of someone "standing" on the surface, a geodesic never turns left or right — illustrating why straight strips naturally follow geodesic paths when laid on a surface.

It is this property of the geodesic that motivates its use in this thesis. Even prior to the publication of Pottman's work [20] for producing strips using families of geodesic curves as input, we were pursuing the approach of covering meshes with geodesics in order to solve this problem. The availability of this useful work affords us the luxury of addressing the strip tiling problem purely by focusing on covering the surface with geodesics and because we can defer to Pottmann [20] for construction of the actual material strips to approximate the surface.

Our geodesic-finding algorithm draws upon much recent excellent work on geodesics for discrete surfaces [4, 14, 23]. Specifically, we make significant use mesh intervals called "windows" which can be unfolded into the plane since they contain no vertices and over

which we can search for geodesics with desirable properties since they are straight when unfolded. Without this wide body of previous work our solution to this problem would not have been possible.

However, these works focus on solving the shortest path problem in order to find geodesics. While all shortest paths are indeed geodesics, not all geodesics are shortest paths [8]. Therefore, in order to find *all* of the geodesics we need to extend these algorithms.

1.3 Applications

If material strips need not be straight when unrolled, or *developed*, into the plane or if the surface that is to be tiled is a "special case" surface (e.g. a section of a sphere, a developable surface, a surface of revolution, etc.) then either existing techniques (e.g. strippification) or surface properties (e.g. symmetry) can be used to compute the lines required to tile the surface.

However, if planar material available *only in rectangular strips of bounded width* must be used to cover an *arbitrary curved surface* then to the best of our knowledge this algorithm is the only method available for tiling the surface. As such, the following applications can make significant use of this work.

1.3.1 Roof surfacing

With the exception of single layered roofs made from exotic materials like glass or fabric, most building roofs have two distinct parts - the supporting structure and the weather resistant surfacing. For curved roofs, which are gaining popularity in modern architecture, both structural and surfacing construction is far more complicated than for that of planar roofs.

The de facto standard material for surfacing commercial sloped roofs are metal roofing panels which are abundantly available in long straight strips. And since these strips are typically joined end to end using longitudinal slip joints that must preserve their straightness in order to allow for thermal expansion slippage along their length, their effective length is infinite.

As mentioned in Section 1.2 straight strips placed on curved surfaces will always follow geodesic paths. Since finding "parallel" geodesics on a surface is far more difficult than

finding parallel lines in the plane this fact vastly complicates the strip tiling problem that is trivial for planar roofs.

Figure 1.2 shows a large curved roof covered with these metal panels. As is evident in Figure1.2(b), the metal strips are trimmed and placed appropriately in order to tile the surface without gaps or overlap. Given a mesh of this surface and the material width, our algorithm can be used to prescribe this trimming and placement.

1.3.2 Boat hulls

Some boat hulls are made from straight flexible materials like thin wooden strips. Figure1.1 shows an example of how this algorithm can be used to produce the strips necessary to form the hull.

1.3.3 Wigs

This algorithm could also be used to create custom fitted wig foundation caps. Using this algorithm hair available in straight strips could be trimmed and joined to custom fit anyone's head. Figure 5.4(a) illustrates this application.

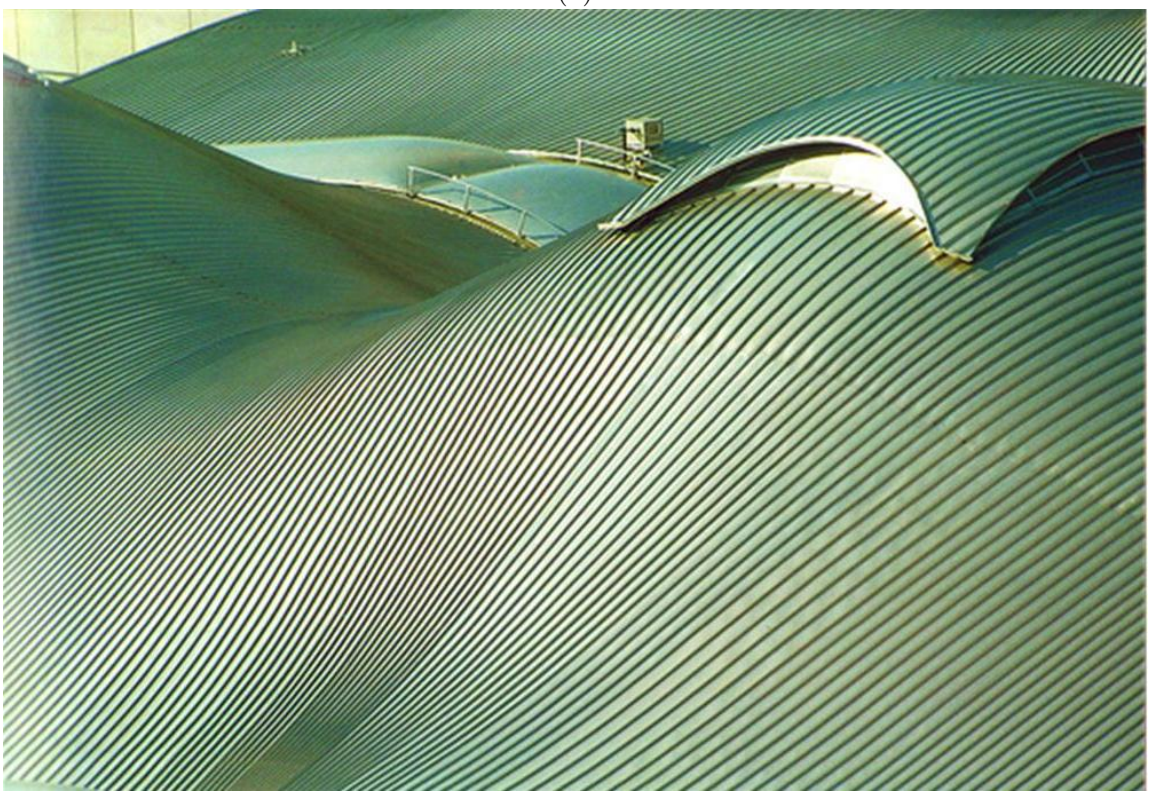
1.4 Contributions

Our contributions in this thesis include the following. First, we adapt existing shortest path techniques in order to be able to find *straight*, as opposed to shortest, geodesics. Then, we develop techniques to satisfy more geodesic curve constraints of greater complexity from a wider range of starting points as compared to the constraints posed by the shortest path problem. Finally, we present a novel geodesic area calculation technique which enables us to choose strip candidates that minimize material waste.

Our algorithm is exact with respect to the polyhedral geometry of the mesh surface and runs on a variety of surfaces with a modest time complexity of $O(n^{1.5})$, where n is the mesh size. The required user-defined orientation typically allows us to incorporate important properties of the surface or application-specific knowledge, e.g. symmetry, gravity, or other considerations, into the algorithm. Finally, we show how our algorithm can be extended by relaxing the constraint that neighbouring geodesics span the mesh. In other words, the delimiting geodesics constructed are allowed to intersect in the interior of the input surface.

This allows straight strip tiling of a wider variety of meshes including those with greater undulation, as we show in Section 5.

(a)



(b)

Figure 1.2: The Southern Cross Station in Melbourne, Australia covered with straight metal strips (photos courtesy of Alan Lam).

Chapter 2

Background

In this chapter we investigate alternative approaches to solving the straight strip tiling problem as well as techniques upon which we build in the course of developing our chosen approach.

2.1 Remeshing and strippification

As mentioned this problem could be solved by remeshing [2] the input surface with small enough facets. However this solution does not take advantage of the available length of the strips and thus leads to an unnecessarily fractured result.

There are also a variety of strippification algorithms [24]. The common issue with them is that they do not address the straightness criterion. Our effort is on finding appropriate straight geodesic curves on an open mesh patch satisfying user and material width constraints.

2.2 Semi-discrete surfaces

Recent work from architectural geometry [11] uses planar quad meshes to represent freeform structures. This work is extended by combining these quads into continuous singly curved strips in what is known as a semi-discrete surface representation [20].

As mentioned in section 1, Pottman et. al. go on to show that if geodesic curve families are used as input then straight strips which approximate the input surface can be produced. In a similar work [21] the authors call for the creation of curve networks to initialize their

semi-discrete surface optimizations which is in fact the objective of this paper. If we are successful in finding such a family of geodesic curves then this existing body of work can be used to approximate the input mesh with a semi-discrete surface made out of singly curved strips that are straight when unrolled.

2.3 Geodesics

Geodesics are most well known for the fact that the shortest path between two points on a smooth manifold is guaranteed to be geodesic [3, 5]. However, they are more rigorously defined as curves whose curvature vector is parallel to the surface normal. There have been many studies extending this concept of differential geodesics from smooth surfaces to discrete geodesics on polyhedral surfaces that are represented by meshes.

2.3.1 Shortest geodesics

Shortest geodesics are defined as the locally shortest curves on a mesh [6, 14]. The most popular method for finding these shortest geodesics on meshes has been using fast marching to solve the Eikonal equation [10]. Using this method, distances from a source point are propagated across the mesh face by face with the shorter distances cancelling out the longer ones.

The other popular approach for finding shortest paths does so by exploiting their *straightness* property. The MMP algorithm [14] originally introduced in 1987 and made practical by Surazhsky et. al. in 2005 [23] divides the mesh into intervals — or *windows* — without vertices over which geodesics exist simply as straight lines in that window's planar unfolding. By creating, propagating and merging appropriately, *straight* windows can be extended from any source point to all other points on the mesh and the shortest path can simply be selected from among those that reach the intended target.

Bommes and Kobbelt [4] extend this work to the computation of distances from general polygonal segments on the surface rather than just single points. Liu et. al [12] warn of *degeneracies* that exist in the Surazhsky algorithm.

2.3.2 Straightest geodesics

Straightest geodesics are those whose left and right curve angles (the sums of the incident angles on either side of a point on the curve on a mesh) are always equal [18]. Straightest geodesics have application to the translation of vectors and the integration of vector fields and are related to shortest geodesics in the following ways [18]:

- A geodesic containing no surface vertex is both shortest and straightest.
- A straightest geodesic through a spherical vertex is not locally shortest.
- There exist a family of shortest geodesics through a saddle vertex. Only one of them is a straightest geodesic.

Saddle and spherical vertices are described fully in section 3.2.

2.3.3 Quasi-geodesics

Despite their popularity, neither of the above discrete geodesics serves our purpose because we are interested in *all straight geodesics* — not just those that are the shortest or the straightest.

Quasi-geodesics are the limit sets of smooth geodesics on polyhedral surfaces [1, 16]. They behave identicaly to shortest geodesics with the exception that there is also a family of quasi-geodesics through a spherical vertex (which as mentioned above cannot contain shortest geodesics [14]). It is this higher adherence with the differential version of the geodesic that makes quasi-geodesics the most suitable for our application.

2.4 Parameterization

Mesh parameterization techniques [22, 7] also offer potential. Specifically, the straight strip tiling problem could be solved if a parameterization could be found that preserved straightness in the sense that straight lines in the parameter domain would correspond to geodesics on the original surface (and vice versa).

Since angle preserving parameterizations minimize local angular distortion between the original surface and the parameter domain [22, 7], they have the most potential for preserving the “straightness” that is necessary to solve the strip covering problem. However, if we try

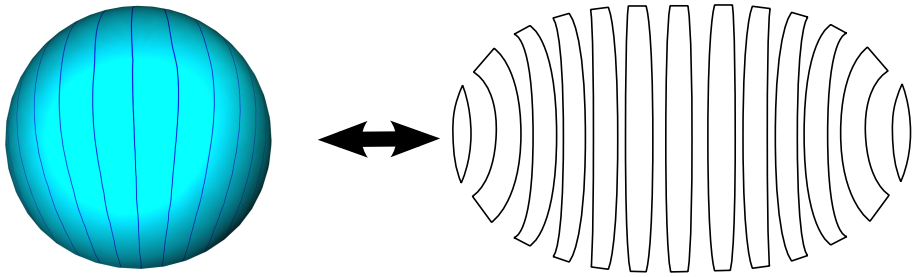


Figure 2.1: A hemisphere parameterized with ABF produces strips that when developed are not straight.

to parameterize a hemisphere using Angle Based Flattening (ABF - see Figure 2.1) [22] we find that when parallel lines in the parameter domain are mapped back onto the sphere and the resulting segments are approximately flattened, they are not straight in that they do not fit well within material rectangles.

In fact, by looking again to the example of the hemisphere it is evident that no single parameterization exists in the common sense for a given surface in that none preserve this straightness *over all user defined orientations*. We see this by first observing that the only geodesics that produce straight strips on a hemisphere are the so called great arcs which must start and end at common poles if they are to avoid intersection (see Figure 2.2). For example, if we use as our hemisphere a vertical bisection of the globe — e.g. the western hemisphere — then lines of *longitude* become our valid tiling lines.

The parameterization that corresponds to this mapping is in fact the *Mercator projection* [13] which maps lines of longitude in the earth to vertical lines on a typical map of the earth. Now if we use this *same parameterization* but choose a different set of lines on our map (which is our parameter domain) — for example the lines of *latitude* — the

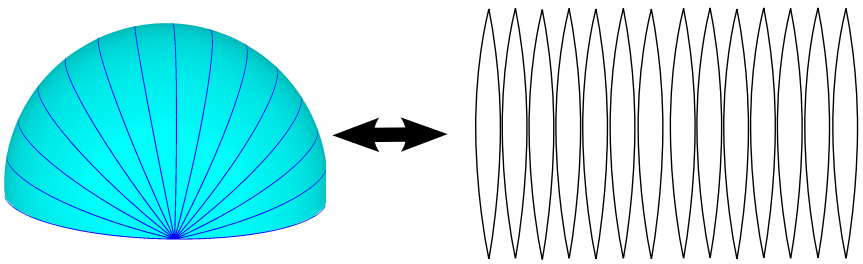


Figure 2.2: A hemisphere tiled along the so called great arcs (with common poles) gives rise to strips that when developed are straight.

resulting tiling lines on the globe are no longer great arcs and are not geodesic. Thus the parameterization of a given surface not only needs to preserve straightness but needs to do so with respect to the user defined orientation.

2.5 Geodesic flow

Another approach, that of the *geodesic flow* [15], also offers potential for solving the straight strip tiling problem. A geodesic flow is a tangential vector field on a surface whereby all resulting integral curves (i.e. curves whose tangents vectors belong to the vector field) are geodesic. If a geodesic flow for a surface could be found then the desired strip boundaries for any material width could be found trivially.

There has been some work on geodesic flows for discrete surfaces [17] describing the properties of fields corresponding to geodesics that emanate from a single point on a mesh. However, application of this theoretical technique to our problem does not appear obvious so we have chosen not to pursue this approach here.

Chapter 3

Straight strip tiling algorithm

As mentioned in Section 1, the work of Pottmann et al. [20] allows us to concentrate exclusively on the creation of geodesic curves rather than the construction of 2D strips from these curves. Our goal then is to seek a family of curves on a given open patch to fulfill all of the following:

- They must be geodesic.
- One curve respects the user-specified point and direction.
- No point on any curve should be more than the material width δ geodesic distance away from either of its neighbouring curves. In other words, a "geodesic circle" with radius δ centered at any point on any curve should intersect another curve or the mesh boundary on either side; see Figure 3.1 for an illustration.
- The curves should be as far away as possible from each other so as to make the best use of the material.
- The curves do not intersect in the interior of the mesh.

Note that there is no guarantee that all the constraints above can be satisfied for any surface with any width bound δ. If we can find such a family of geodesic curves, then we can pass them to the strip generation algorithm of Pottman et al. [20] to produce the straight strips to tile the input mesh. When the input surface is sufficiently curved with a sufficiently small δ bound, the last constraint above can be impossible to satisfy. In Section 3.5, we present an extension to our core algorithm which allows long strips to be "severed"

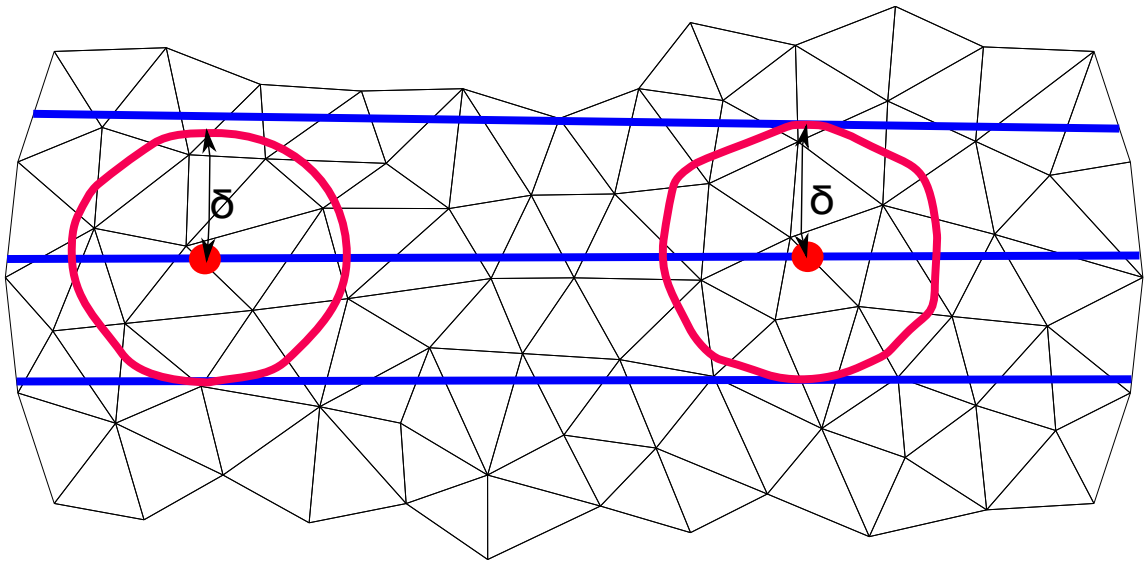


Figure 3.1: Geodesic circles of radius δ illustrating the width constraint test for two points (red dots) on the middle curve. The point on the right passes the test since the circle intersects another curve on either side while the point on the left does not. Every point on every curve must pass this test.

by allowing constructed geodesics to intersect. This way, we are able to handle surfaces with greater undulation.

3.1 Source geodesic curve

We first create the initial curve from the user-specified surface point p and tangent vector $\vec{v}$. Since our curve is to be geodesic, we can simply start at the given point and proceed "straight" along the surface in the given direction until we reach the boundary. To do this, we utilize the classical definition of a geodesic as a path whose osculating plane also contains the surface normal [5]. In the discrete version, a geodesic proceeds straight over faces and preserves incident angles when traveling over edges [18]; traveling directly over vertices will be discussed below in Section 3.2. Using this simple rule we implement a *geodesic walk* that takes as input a starting point and tangent vector and simply "walks" forward producing a geodesic curve as output.

3.2 Constrained neighbour geodesic construction

Given this source geodesic, we implement a method for finding a neighbouring geodesic that satisfies the constraints mentioned above. Then we use this method recursively to find subsequent neighbours until we reach the boundary of the mesh patch. Finally, we repeat this recursive process on the other side of the source to find the rest of the geodesics.

We satisfy our width constraint by appealing to the work of Bommes and Kobbelt [4] for finding geodesic distance fields from polygonal segments on meshes to find the boundary of the region locally "swept out" by our geodesic circle of Figure 3.1 as its center travels along the source curve. Since this resulting connected *upper bound curve* represents the furthest extent of all of our geodesic circles, we can use it to *simultaneously enforce the width constraint test at all points along the source curve* by simply requiring that our neighbouring geodesic *not cross* this upper bound.

Moreover, since we know this upper bound is continuous (i.e. connected), the width test is also enforced in the other direction. I.e., a test centered at any point on a neighbour curve within the upper bound will always pass because the upper bound's continuity guarantees that one of our circles "swept over" that point during the construction of the upper bound ensuring the existence of a short enough geodesic to a point on the source.

This upper bound curve is constructed by partitioning the mesh into two different types of vertex-less regions (called windows - see Section 2.3.1) that extend perpendicularly from the source: *linear windows* and *vertex windows*. A linear window contains an entire segment of the source curve and in the unfolding of the window's faces the boundary of the region swept out by our geodesic circle is easily computed as a line segment parallel to the source.

Vertex windows are required because a geodesic directly incident upon a vertex may continue along *any* ray between the two rays at an angle π from the incident ray [23] (Figure 3.3) and still retain its geodesic properties (if this were not the case there would be regions of the mesh unreachable by geodesics from a single source). This transition from single geodesic ray to angular range splits the incoming window at the vertex and also gives rise to a new window propagating radially from this new or *pseudo*source.

In the unfolding of vertex window faces, the extent of the geodesic circle test is simply a circular arc. Figure 3.2 shows these linear and vertex windows (separated by dashed lines) and illustrates how the resulting upper bound curve is simply a connected set of the aforementioned geodesic and arc segments.

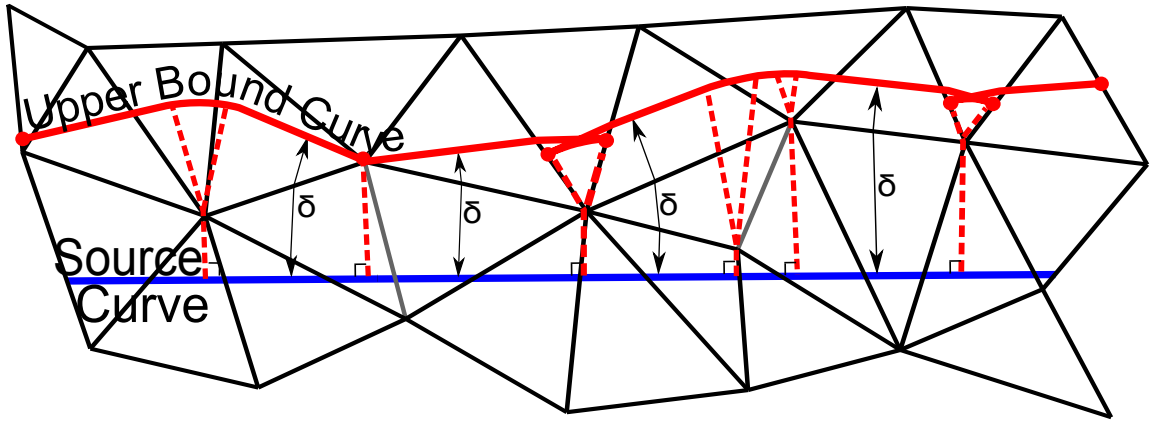


Figure 3.2: Given an initial source geodesic curve (bottom in blue) we form the local boundary of our width constraint test circles (Figure 3.1) at δ (which we call the upper bound - top in red) by extending windows (vertex-less regions - separated by dashed lines) from the source curve and measuring distance in the planar unfolding the window's faces. The upper bound is made up of straight (i.e. geodesic) and circular arc sections - with convex corners (i.e. those turning away from the source) marked with red dots.

Our need to create vertex windows for *both* saddle (face angle sum $> 2\pi$ - also known as "hyperbolic") and spherical (face angle sum $< 2\pi$) vertices arises from our need to use quasi-geodesics (which in turn arises from our need to find all straight geodesics as described in Section 2.3.3). The resulting implementation differences from previous work on finding discrete geodesics [4, 23] are described in Chapter 4. This difference also simplifies area calculations as described in Section 3.3.

As mentioned, in order to ensure that the width test passes at all points on the source *and* the neighbour, we need only ensure that our neighbouring geodesic *does not cross the upper bound*. Furthermore, we can interpret our constraint that neighbouring curves be as far away as possible from each other to mean that at least one point on the neighbouring geodesic *must touch the upper bound*. If this were not the case, we greedily claim that our neighbour would not be far enough away from the source and thus we would not be making the best use of material. The possibility that a neighbour not touching the upper bound could in fact be the optimal choice seems unlikely; however, formal confirmation of this fact is left to future work.

Armed with this knowledge that our neighbour must touch, but not cross, the upper bound we consider the upper bound curve's shape (Figure 3.2). In the face of (or in the

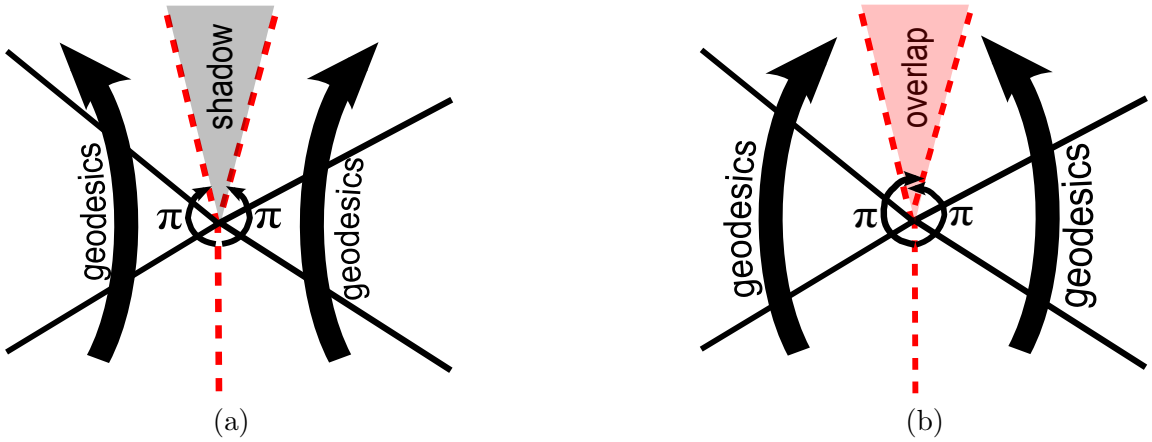


Figure 3.3: The window boundaries surrounding a saddle ((a) - where the face angle sum $> 2\pi$) and a spherical ((b) - where the face angle sum $< 2\pi$) vertex are shown separated by dashed lines. A vertex splits the incoming window into two windows as well as giving rise to a third window emanating from this new or *pseudo*source. For the purposes of finding *straight* (as opposed to shortest path) geodesics, *both* saddle *and* spherical vertex behaviour must be defined in this way.

unfolding of the faces local to) a point along the upper bound curve, we can view our point as a point on a curve in the plane. In the plane, the only points on a curve that a line segment (which is a geodesic unfolded) can touch but not immediately cross the curve are where the curve *curves away* from the line segment. We call these points *convex corners* (red dots in Figure 3.2) of the upper bound. We conclude that all valid candidate geodesics *must touch at least one convex corner* of the upper bound.

We can use this fact to enumerate all potential candidate geodesics by observing that all geodesics touching one convex corner must lie within double-sided angular ranges, or "butterfly"s, emanating from this convex corner (see Figure 3.4). We can then enumerate all the geodesics within these butterflies by segmenting them into vertex-less windows within which all geodesics exist as straight lines in the window's unfolding. Finally, in order to satisfy the width and self-intersection constraints, we discard any of these geodesics that cross back over the source or upper bound curves.

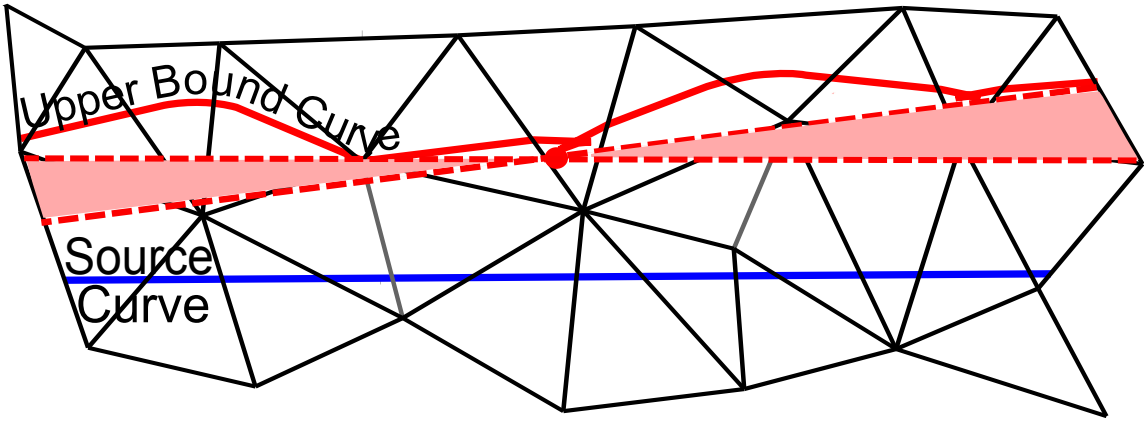


Figure 3.4: The dashed lines delimit a "butterfly" window (a pair of opposing angular ranges) emanating from a convex corner point (red dot - where the upper bound curves "away" from the source) of the upper bound curve that contains all candidate geodesics touching that point.

3.3 Avoiding material waste

While different applications may have unique criteria against which to judge and select the best valid candidate neighbouring geodesics, one that is likely to have universal application is that of material usage maximization. Specifically, among all the valid candidates in one recursive step, we greedily select the one with the most *usage* - measured as the enclosed surface area divided by the source curve length.

The enclosed area between the source and candidate curves can be easily found by summing the areas of the enclosed faces. However, since in the unfolding of a window's faces all geodesics are *straight lines*, we can apply one such summation to all other geodesics in the same window by simply adding or subtracting a triangle (Figure 3.5). Moreover, if successive windows are alongside (i.e., in full contact with) one another, we can propagate this area to the next window. Finally, we can even propagate this area calculation from butterfly to butterfly by exploiting the fact that the last geodesic of one butterfly is actually the first geodesic of the next.

Fortunately, since we are using quasi-geodesics (Section 2.3.3), the windows emanating from a source point are always alongside each other. This is because quasi-geodesics travel across spherical vertices similarly to saddle vertices in that geodesics incident upon them may continue along any ray between the two rays at an angle π from the incident (Figure 3.3) as

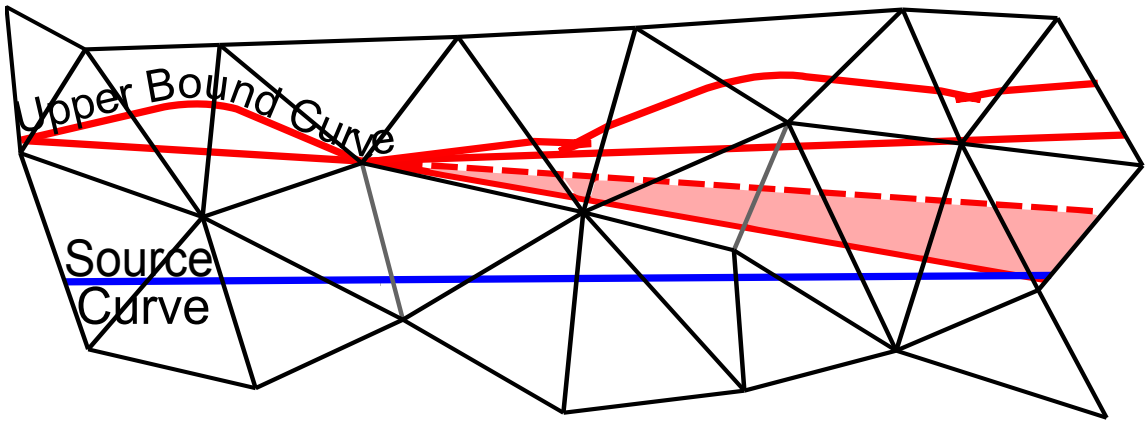


Figure 3.5: Given the area enclosed by one geodesic (red dashed line) in a window (delimited by solid red lines), the area enclosed *by any other* geodesic in that window can be found by adding or subtracting the triangle between them (pink shading) in the unfolding of the window's faces.

opposed to the previous work on finding discrete geodesics [4, 23] in which spherical vertices are ignored as shortest paths never travel directly over them [14]. These spherical vertex windows (the pink overlap region in Figure 3.3 (b)) bridge the "gap" between the windows on either side of the vertex ensuring that all windows are alongside one another. We will see in Chapter 4 how to extend these discrete geodesic works [4, 23] to find quasi-geodesics as opposed to shortest geodesics.

3.4 Computational complexity

In the first stage of the algorithm the initial geodesic is constructed using the geodesic walk algorithm described in Section 3.1. To do so the algorithm must "walk" over $O(F)$ faces (with each face requiring constant time) for each of the $O(\frac{BoundingBoxSize}{\delta})$ geodesic curves that are to be output.

The second stage of the algorithm involves the recursive construction of a neighbouring geodesic given a source geodesic. This stage includes the following steps:

1. Windows are extended from the source geodesic to construct the upper bound curve (Figure 3.2). This is done by creating one window for every face crossed by the source curve and then propagating these windows forward - splitting each one into *three* when a vertex is encountered (Figure 3.3).

2. At *each convex corner* (Figure 3.4) of the upper bound butterfly windows are constructed and searched for the geodesic enclosing the greatest area (Figure 3.5).

3. A geodesic curve is output. Since geodesics travel straight over faces only the intersections with the $O(E)$ edges need be stored. Again we expect $O(\frac{BoundingBoxSize}{\delta})$ geodesics to be output.

Thus step 1 includes construction of $O(F + 2V)$ windows (where V is the number of vertices) each of which propagates over $O(F)$ faces (when summed over all strips on the mesh). In step 2 we encounter $O(V)$ convex corners (again summed over all strips) each of which gives rise to $O(V)$ windows each containing $O(F)$ faces. Finally in step 3 the output size is $O(\frac{E\,BoundingBoxSize}{\delta})$. Hence the theoretical upper bound is $O(n) + O(n^3) + O(\frac{(F+E)BoundingBoxSize}{\delta})$ or $O(n^3)$ (where $n = max(V, F, E)$) as long as δ is not much smaller than the average edge length.

Note that using the area calculation propagation method in Section 3.3 requires constant time for each window and thus does contribute to complexity. Also note that we do not require the $O(\log V)$ steps needed to perform the window sorting required by shortest path algorithms [23] (see Section 4).

However on a sufficiently smooth mesh, the butterfly windows from step 2 do not overlap appreciably which means that there should be approximately $O(V)$ windows within all of the butterflys over the entire mesh. Moreover, each of these windows need propagate over only the width of the patch which should only be $O(\sqrt{F})$ faces on a uniform mesh that is reasonably "square" (i.e. width and length are of the same order).

Under these assumptions our complexity becomes $O(n^{1.5})$ (again where $n = max(V, F, E)$ and δ is large enough). It is important to note that this argument does not constitute a rigorous "average case" complexity analysis but rather a best guess at actual expected performance of our algorithm on a specific subset of the input parameters (i.e. smooth, uniform, square meshes with a material width that is not much smaller than the average mesh edge length).

3.5 Strip severing

The algorithm as described thus far attempts to cover an open mesh with non-intersecting geodesics. Unfortunately, especially for surfaces with high curvature, this is not always

possible. However if we allow these partitioning geodesics to touch each other we can remove this limitation and extend the algorithm's applicability to any surface on which a valid source geodesic can be found. I.e., if a source geodesic starting from and ending on the mesh boundary that does not intersect itself can be found then using this technique our algorithm will succeed.

This amounts to relaxing the constraint that candidate geodesics not intersect the source geodesic curve. In other words, candidate geodesics with one or both endpoints on the source geodesic are now also considered. And when we select one of these segment candidates as our greedy choice, our source curve in the next iteration will be a connected set of geodesic segments rather than a single geodesic curve — with the corresponding upper bound curve being simply the union of each geodesic segment's upper bound (which *should* always meet due to the fact that these geodesic segments must always join at convex angles — proof left to future work). Figures 5.1 through 5.6 show the straight strip tiling results of using this technique.

Chapter 4

Implementation and window propagation

The straight strip tiling algorithm was implemented using C++ and DirectX following closely the published works of Surazhsky et. al. [23] and Bommes and Kobbelt [4]. While all of the interesting details of our implementation are beyond the scope of this paper, *window propagation* details are the most notable differences from and additions to these previous works.

Window propagation is the process of constructing windows (Section 3.2) starting from a source point or line segment and preceding forward face by face. In our implementation windows are propagated from the source curve to form the upper bound as well as from butterfly windows (Figure 3.4) to find candidate neighbour geodesic curves.

The objective of both of these window propagations is to find *all* geodesics (or quasi-geodesics — see Section 2.3.3) present. This differs from the previous work [23, 4] that seek only the shortest path geodesics. As such, our implementation contains the following extensions:

- As mentioned in Sections 3.2 and 3.3, instead of being ignored, spherical vertices give rise to windows emanating from between the two rays at an angle π (in either direction) from the incident ray (Figure 3.3(b)).
- Window merging, used to eliminate non-shortest path windows [23, 4], should not be done as it eliminates valid geodesics.

- Care must be taken when backtracking (following a window back to the source) over vertices since the simple rule of selecting the shortest path back to the source no longer applies. Instead, any back-path between the two rays at and angle π from the incident back-path are potentially valid. Either application dependent rules (e.g. always use highest or lowest) or back-path tracing (i.e. leaving a "trail of breadcrumbs" over vertices which we did) needs to be used to choose between more than one valid back-path.

- While in shortest path algorithms *boundary vertices* give rise to new pseudo-sources to allow shortest paths to curve around edges of the mesh, during butterfly window propagation boundary vertices should be ignored because candidate curves must be geodesic.

- On the other hand, during source curve propagation, boundary vertices require special processing. Since we do not propagate circular distance windows from the source curve's end points as do Bommes and Kobbelt [4] (because our source curve effectively has no end points), our computed upper bound may not always reach the boundary of the mesh. This happens when the mesh boundary proceeds "outward" from the end of the source curve.

  To resolve this problem, when a boundary vertex is encountered during window propagation we unfold the boundary faces into the plane searching for the first point along the boundary that is a distance δ from the line supporting the source geodesic. Then we simply attach this boundary point to the end of our foreshortened upper bound curve.

- As mentioned in Section 3.4, windows do not need to be sorted and propagated in order of length since we are not looking for the shortest path. Instead, windows need to be created and kept in sequential order so that a connected upper bound can be constructed and also so that area summations can be propagated from window to adjacent window.

- Rather than an interpolated estimate combined with triangle decimation as in Bommes and Kobbelt [4], we need to find an exact iso-distance curve when forming our upper bound. Fortunately since our algorithm does not actually use the much harder to construct arc sections of this iso-distance curve (see Section 3.2 and Figure 3.2), we

replace them with lines segments making calculation of the exact iso-distance curve easier.

Aside from these differences, windows are propagated as described in the previous works mentioned.

Chapter 5

Results

We now present straight strip tiling results produced by the implementation of our algorithm, examining different factors that influence such results and report timing as well as limitations. We consider three factors: the width bound δ, the initial curve orientation p, $\vec{v}$, and surface undulation or range of curvature.

A set of additional results are shown in Figures 5.4, 5.5 and 5.6 with the corresponding timing and material usage statistics is provided in Table 5.1. When reporting timing, we do not vary $\vec{v}$ or δ as we recall from Section 3.4 that the algorithmic complexity depends only on the number of faces and number of (spherical) vertices in the input mesh. Figure 5.7 compares the observed run times to the average algorithmic time complexity we claim in section 3.4.

5.1 Undulation

Surface undulation appears to have the greatest effect on tiling results, as shown in Figure 5.1. Our experiments show that while p, $\vec{v}$ and δ are held constant increasing the surface undulation gives rise to a relatively abrupt transition from un-severed tilings as in (a) to highly severed tilings as in (b). Note that while the small triangles in (b) appear large, the width constraint is in fact satisfied as every point on the neighbouring geodesic is within a geodesic distance δ from a point on its source.

5.2 Orientation

With the surface and the width bound δ held constant, we observe that tilings are also relatively sensitive to the initial curve orientation p, $\vec{v}$. As shown in Figure 5.2, when the same geodesics are forced to travel through multiple regions of extreme curvature they are more subject to severing than when these extremes are "spread out" amongst the geodesics.

It is also interesting to note that even though the altered orientation is noticeable in the top middle portions of (a) and (b), the algorithm's severing process in (b) in effect *reverts the orientation back* towards the more favourable one in (a). It is not clear whther this is to be expected or is merely a coincidence as finding a favourable orientation was not a goal we hoped to achieve in this version of our algorithm.

5.3 Material width

As Figure 5.3 shows, once a tiling without severances is found reducing δ does not necessarily induce severing as might be expected. Intuitively, this independence may be due to the expected existence of a continuum of non-intersecting geodesics between any two "parallel" geodesics (non-intersecting geodesics no more than δ geodesic distance away from each other) on a sufficiently smooth surface. In fact, if the two geodesics belong to the same geodesic flow, then this is exactly the case. However, exploring the applicability of geodesic flows to the tiling problem is left for future work.

5.4 Limitations

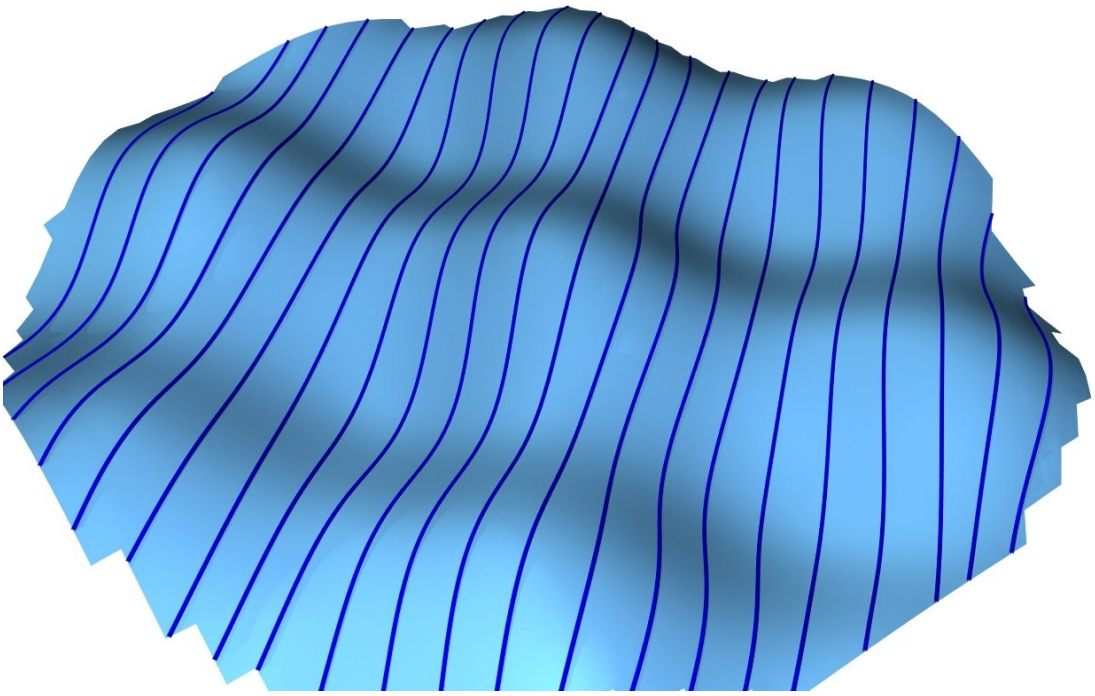
Since the algorithm requires an initial, finite geodesic that spans the mesh, it does not work on closed surfaces. However, if a small hole is punched and a non-self-intersecting initial geodesic can be found that starts and ends at the hole boundary, then the algorithm should succeed under reasonable parameters. Also, since there is no provision in the algorithm for joining separated upper bound curves, it does not work on surfaces that contain holes. An effective means of handling both limitations is to first segment the given surface into open patches with simple boundaries before tiling.

Finally, we emphasize again that creating actual planar strips from the geodesics curves we obtain that can be joined to approximate an input surface with bounded error is a complex optimization problem which has been the subject of recent work [20]. The planar

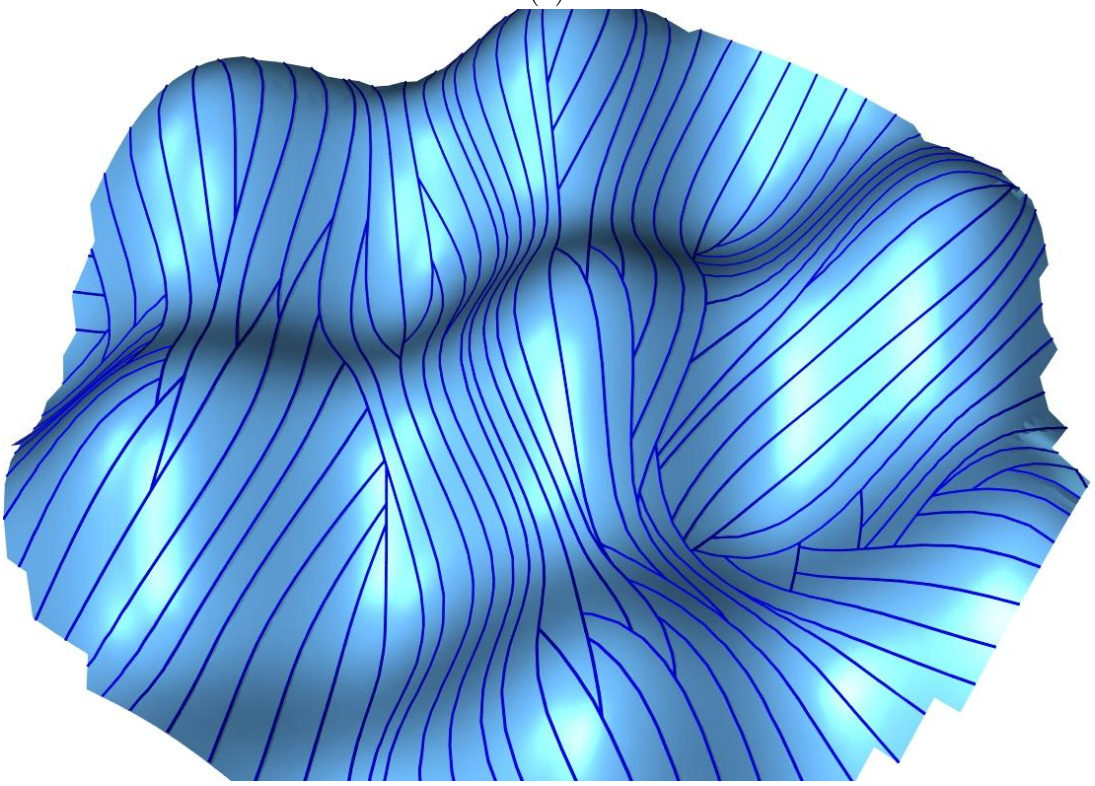
Mesh	**Faces**	**Time**	**Material Usage**
Skull	1,000	<1	71 %
Hood	1,000	<1	86 %
Hat	4,600	3	59 %
Taller hat	4,600	3	65 %
Face	6,100	4	66 %
Ship hull	7,800	5	79 %
Plane fuselage	9,700	15	81 %
Synthetic roof	19,600	20	75 %

Table 5.1: Numerical results of our experiments. All tests were run on a 2.60 GHz Pentium 4 with 2 GB of RAM. Timing results are reported in seconds.

(yellow) strips shown in Figure 1.1(b) are merely meant to indicate the geodesic widths (over the input mesh) of the surface strips. While these plots primarily show that our computed geodesics satisfy the prescribed width constraint, they are also intended to suggest how straight strips of material may be trimmed and joined to approximate the original surface. However it is important to note that the shapes in Figure 1.1(b) themselves *cannot* be joined together to approximate the surface.

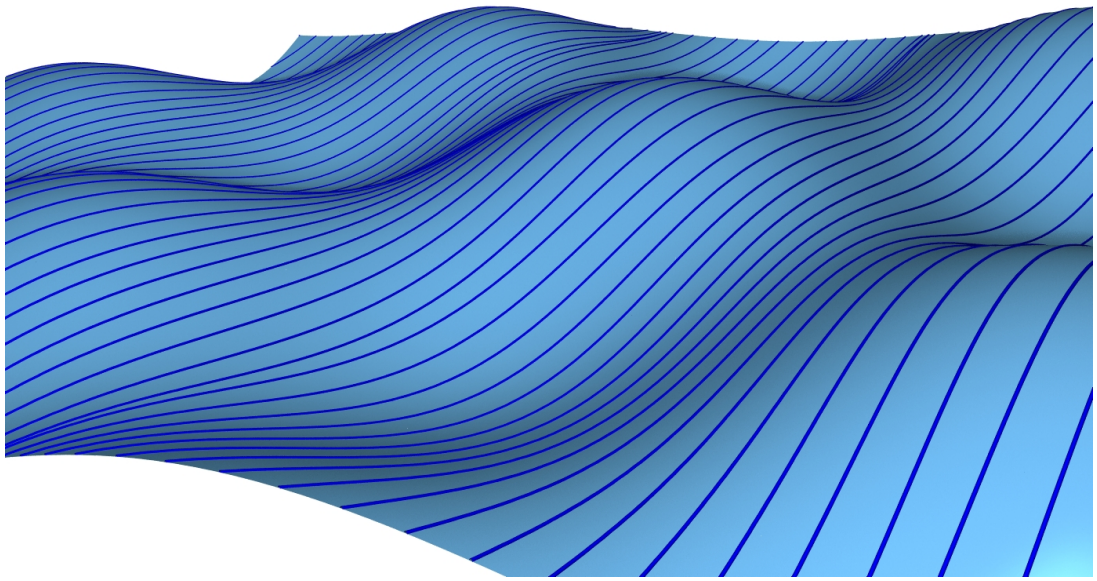


(a)

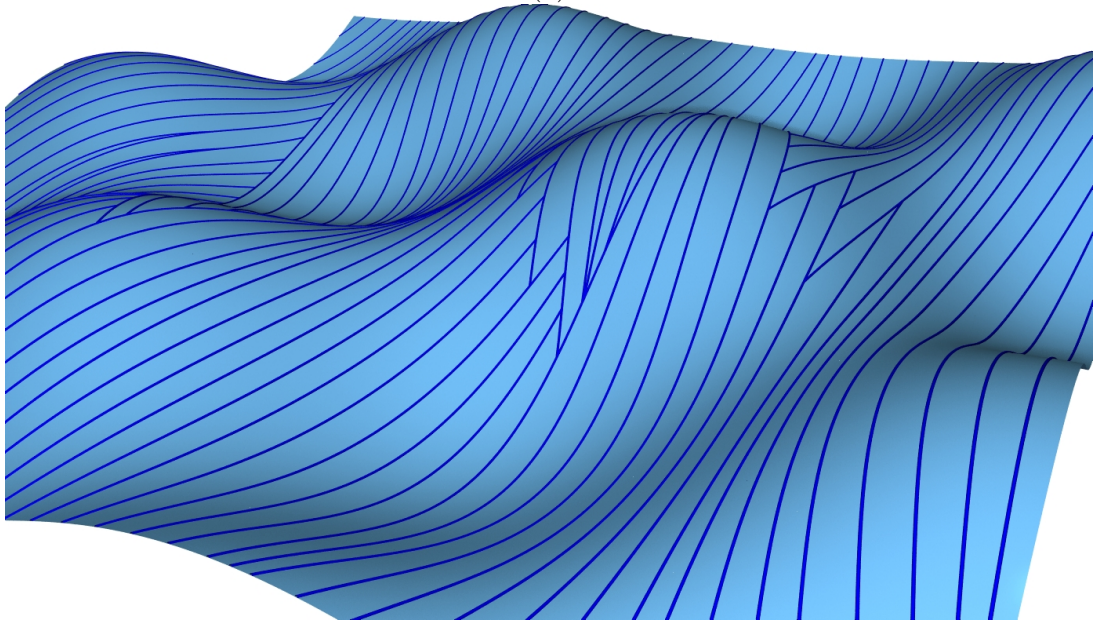


(b)

Figure 5.1: Effects of different factors on straight strip tiling for a synthetic surface mimicking the roof of the Southern Cross Station (Figure 1.2). Increased surface undulation in (b), holding $\vec{v}$ and δ fixed, causes the algorithm to sever tile lines (Section 3.5).

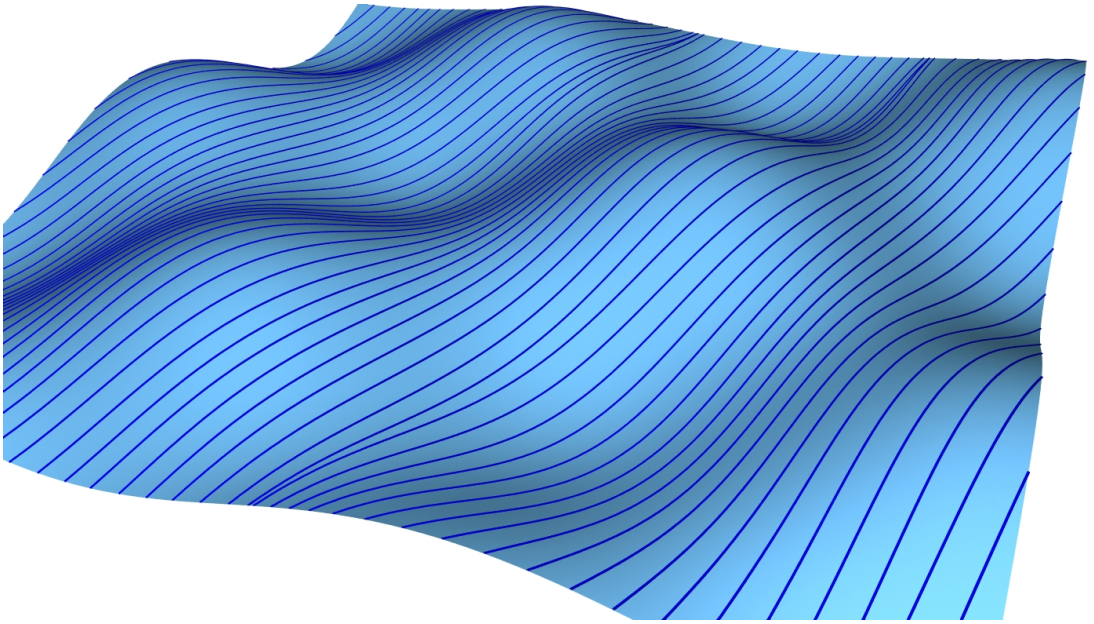


(a)

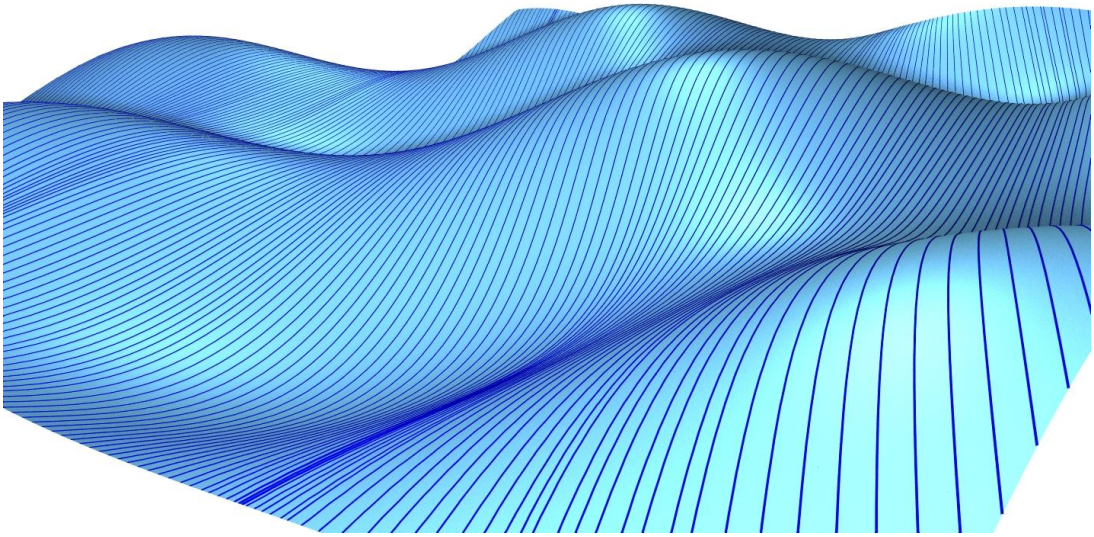


(b)

Figure 5.2: On the same surface and holding δ constant, a change in the curve orientation $\vec{v}$ again causes severing (b) as the geodesics are forced to travel through multiple regions of high curvature.

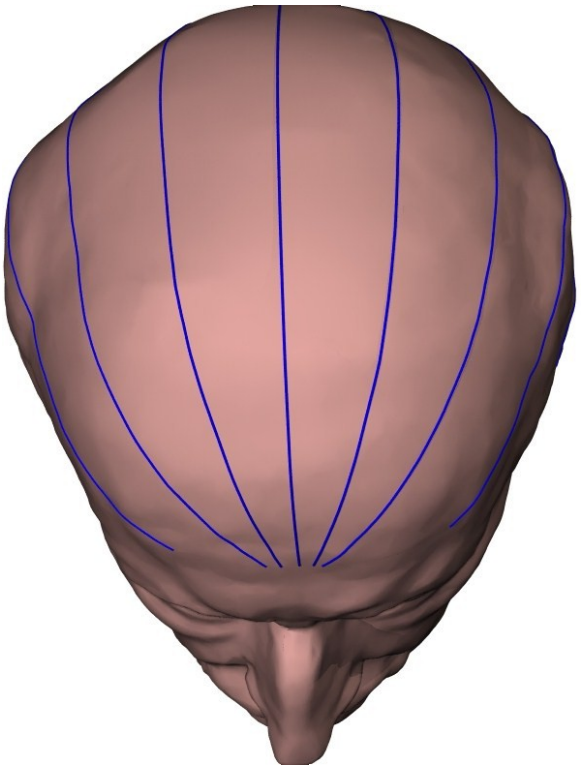


(a)

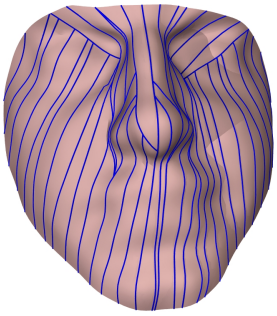


(b)

Figure 5.3: On the same surface while holding $\vec{v}$ unchanged, reducing the width bound δ in (b) does not introduce severing.

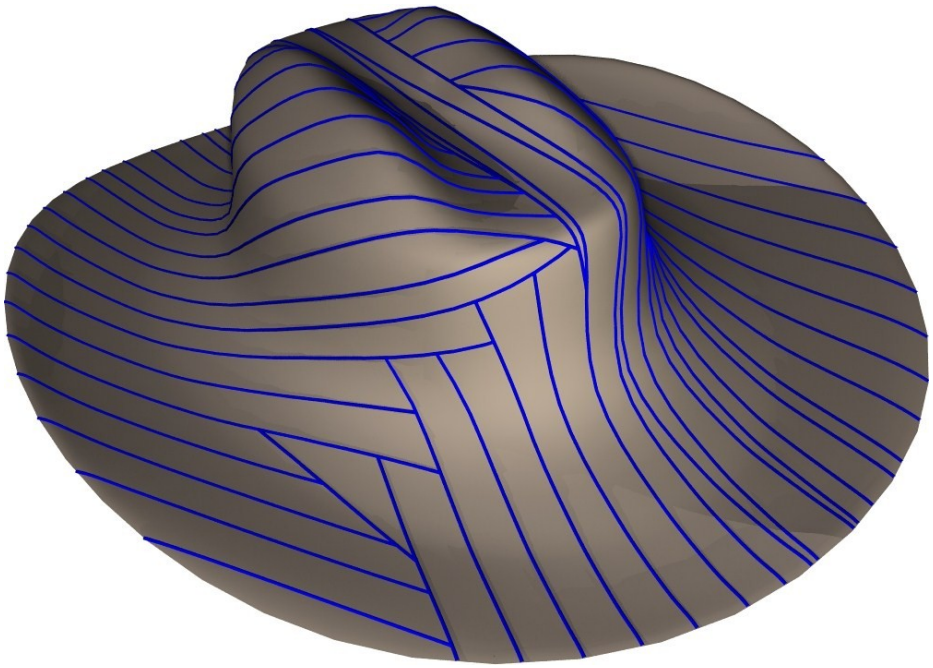


(a) Skull of the Max Planck.

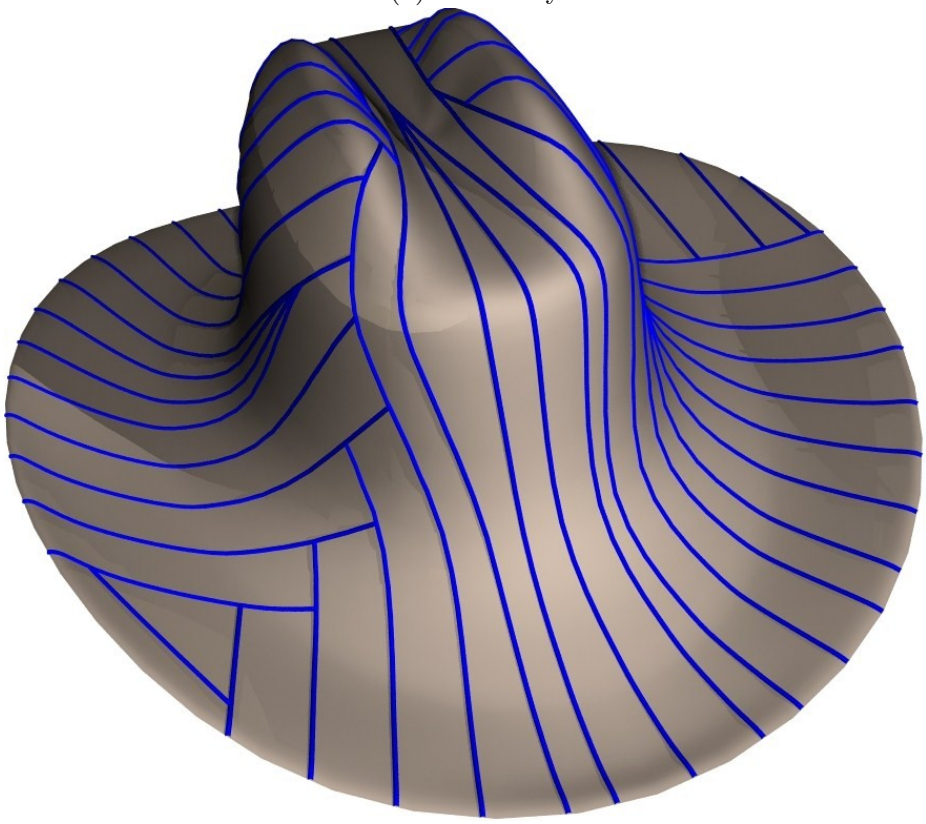


(b) A face mesh.

Figure 5.4: A gallery of straight strip tiling results. See Table 5.1 for run times and material usage.

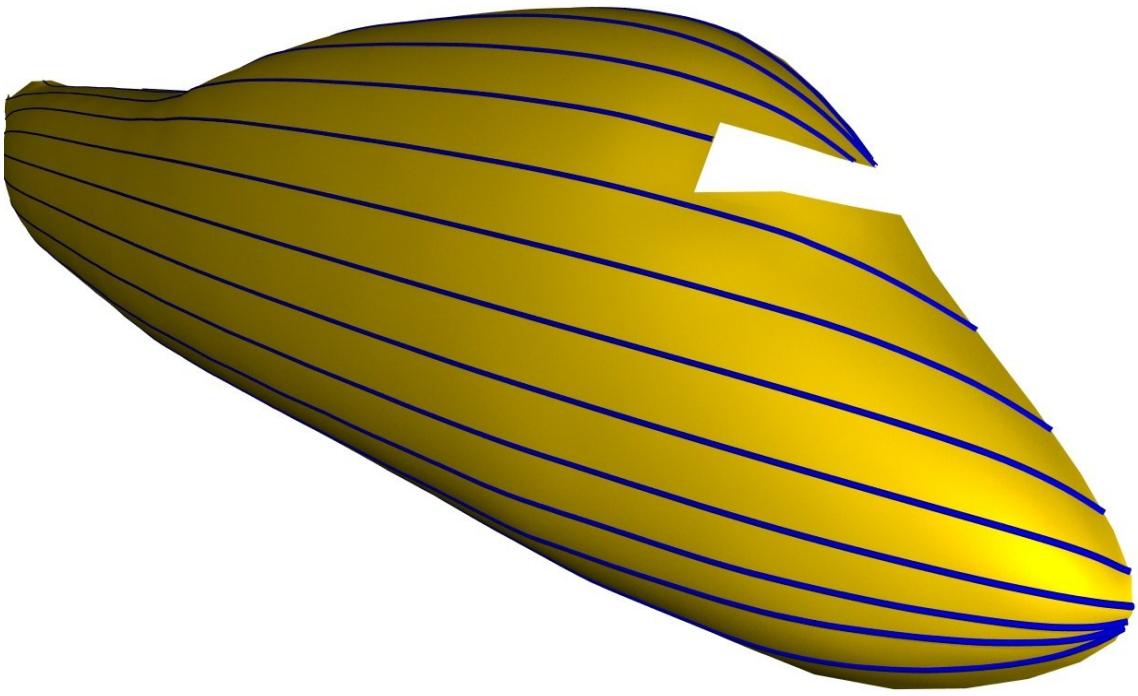


(a) A cowboy hat.

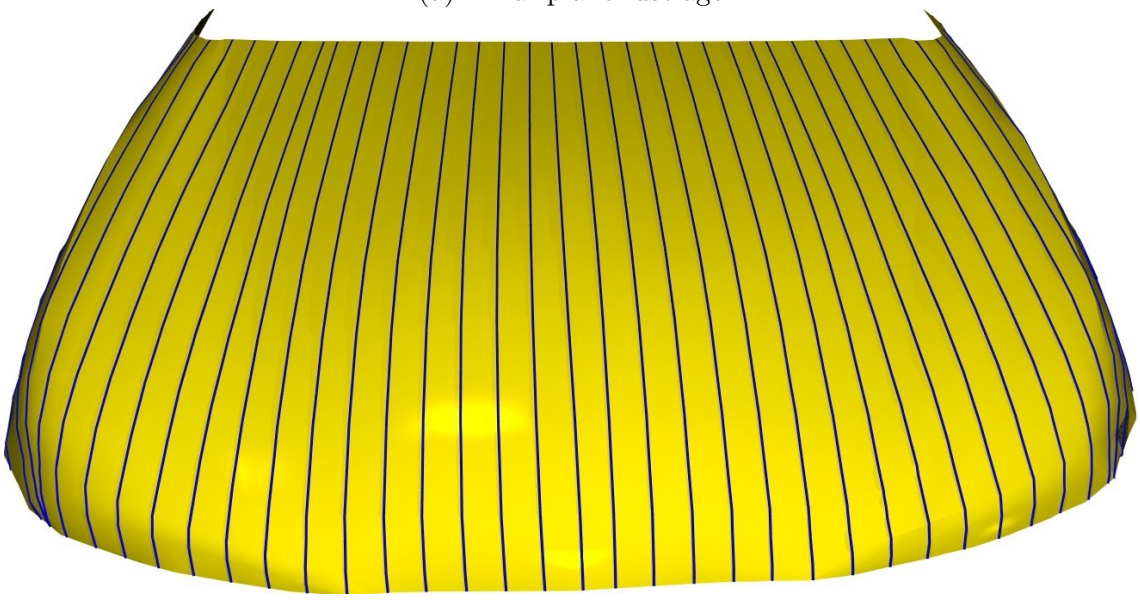


(b) A “taller” hat.

Figure 5.5: A gallery of straight strip tiling results. See Table 5.1 for run times and material usage.



(a) An airplane fuselage.



(b) Hood of a car.

Figure 5.6: A gallery of straight strip tiling results. See Table 5.1 for run times and material usage.

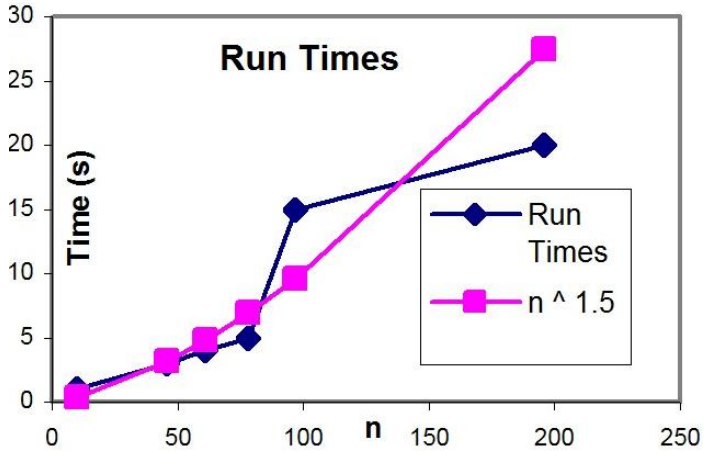


Figure 5.7: Run times from Table 5.1 plotted against the claimed algorithmic complexity of $O(n^{1.5})$ from Section 3.4. A reasonable correspondence exists with two outliers, the airplane fuselage and the synthetic roof. Their deviation from the norm is due to the large difference in the proportion of spherical vertices on these models. Material Usage is estimated as the surface area enclosed by neighbouring strips divided by the area of material consumed (which is the length of the neighbouring curve multiplied by the material width).

Chapter 6

Conclusion and future work

To the best of our knowledge, the straight strip tiling problem tackled in this paper is new. The algorithm proposed is only a preliminary attempt and it still leaves much work to be done. Although the geodesic construction component of the algorithm is exact with respect to the polyhedral geometry of the input mesh and the width bound is also respected exactly, our material optimization is a locally greedy choice rather than a global optimization based on a formal objective function.

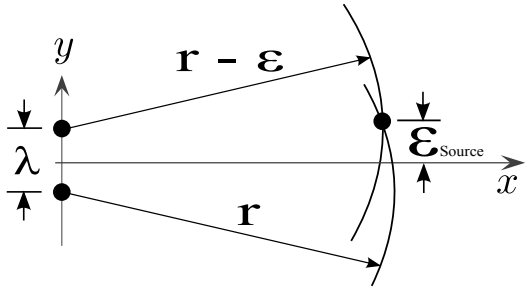
The important question of when the algorithm would fail is also left unanswered. It is conceivable that certain curvature bounds could be found to address this issue. The question of whether there exists a "straightness-preserving" mesh parameterization scheme and the applicability of geodesic flows to the mesh setting to solve our tiling problem are both interesting. Finally, we would like to remove the need for user initialization, seek an optimal general orientation for strip placements, and handle more complex mesh boundary conditions arising from input surfaces with holes.

Appendix A

Degeneracies

Liu et. al [12] warn of "degeneracies" that exist in the Surazhsky algorithm [23]. In particular, they report a number of geometric coincidences whereby window edges exactly coincide with mesh vertices giving rise to a large decision tree. However, they point out that if *exact arithmetic* is used these cases should not happen. They also report window merging degeneracies but these do not apply to us since we do not perform window merging (Chapter 4).

This raises a relevant issue regarding the need for double-precision (i.e. 8-byte floating point) arithmetic when performing geodesic calculations. As an example, consider the triangulation method described by Surazhsky [23] to compute the window source point location from the window end points. The figure to the right shows a round-off error ϵ in one of the triangulation radii r used to triangulate the source of a window of size λ. Using the equations of the arcs, we can solve for the increase in round-off error as:



$$\frac{\epsilon_{Source}}{\epsilon} > \frac{r}{\lambda} \tag{A.1}$$

That is, the error grows according to the ratio of the window's length to its width. Bearing in mind that windows split at every vertex, large ratios on the order of 10^5 or 10^6 or even higher are conceivable as windows propagate across dense meshes - especially if the

source is at an oblique angle to the window unlike in the example shown.

If losing of five or six digits of precision is problematic when using double-precision arithmetic, it is catastrophic when using single-precision (4-byte) arithmetic - the strongly enforced standard in many 3D modeling/rendering environments. Because of this triangulation example and countless other similar examples (not the least of which is area calculation propagation) double-precision is a must.

This example also shows that triangulation may not be the best technique when the point to be located is relatively far away from the triangulation points. However it is not clear how much better our choice of using Cartesian co-ordinates aligned with the window instead of triangulation distances fares since the repeated co-ordinate transformations also introduce significant accumulation of round-off error.

In the end, through careful use of double-precision arithmetic and by directly handling window-on-vertex cases that are not degenerate but are in fact expected, we did not experience the rate of degeneracies reported by Liu et. al [12]. Nevertheless we did detect a non-trivial amount of degeneracies — especially on meshes laid out on a regular grid.

Bibliography

[1] A. D. Aleksandrov. *Intrinsic Geometry of Surfaces*. American Mathematical Society, January 1967.

[2] Pierre Alliez, Giuliana Ucelli, Craig Gotsman, and Marco Attene. Recent advances in remeshing of surfaces. Research report, AIM@SHAPE Network of Excellence, 2005.

[3] Marcel Berger. *A panoramic view of Riemannian geometry*. 2000.

[4] David Bommes and Leif Kobbelt. Accurate computation of geodesic distance fields for polygonal curves on triangle meshes. In *VMV*, pages 151–160, 2007.

[5] Manfredo P. Do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice-Hall, 1976.

[6] E. Dijkstra. A note on two problems in connection with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[7] M. S. Floater and K. Hormann. Surface parameterization: a tutorial and survey. *Advances in Multiresolution for Geometric Modelling*, pages 157–186, 2005.

[8] L. Grundig, L. Ekert, and Moncrieff E. Geodesic and semi-geodesic line algorithms for cutting pattern generation of architectural textile structures. *Proc. ASIA-PACIFIC Conference on Shell and Spatial Structures*, 1996.

[9] L. Grundig, P. Singer, D. Strbel, and Moncrieff E. High-performance cutting pattern generation of architectural textile structures. *IASS-IACM 2000, Fourth International Colloquium on Computation of Shell And Spatial Structures*, 2000.

[10] R. Kimmel and J. Sethian. Computing geodesic paths on manifolds, 1998.

[11] Yang Liu, Helmut Pottmann, Johannes Wallner, Yong-Liang Yang, and Wenping Wang. Geometric modeling with conical meshes and developable surfaces. *ACM Trans. Graph.*, 25(3):681–689, 2006.

[12] Yong-Jin Liu, Qian-Yi Zhou, and Shi-Min Hu. Handling degenerate cases in exact geodesic computation on triangle meshes. *Vis. Comput.*, 23(9):661–668, 2007.

[13] G. Mercator. *Nova et aucta orbis terrae descriptio ad usum navigantium emendate accomodata (New and Accurate Description of the Terrestrial Globe, Amended to Suit the Uses of Navigation).* Duisburgum Doctum, 1569.

[14] Joseph S. B. Mitchell, David M. Mount, and Christos H. Papadimitriou. The discrete geodesic problem. *SIAM J. Comput.*, 16(4):647–668, 1987.

[15] Gabriel P Paternain. *Geodesic Flows.* Birkhauser, 1999.

[16] Aleksei. V. Pogorelov. Quasi-geodesic lines on a convex surface. *Mat. Sb. (N.S.), 25(67), pages 275-306, 1949. English translation: American Math Society Translation 74, 1952.*

[17] Konrad Polthier and Markus Schmies. Geodesic flow on polyhedral surfaces. In *Proceedings of Eurographics-IEEE Symposium on Scientific Visualization 99*, pages 179–188, 1999.

[18] Konrad Polthier and Markus Schmies. Straightest geodesics on polyhedral surfaces. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Courses*, pages 30–38, 2006.

[19] Helmut Pottmann, A. Asperl, M. Hofer, and A. Kilian. *Architectural Geometry.* Bentley Institute Press, 2007.

[20] Helmut Pottmann, Alexander Schiftner, Pengbo Bo, Heinz Schmiedhofer, Weinpeng Wang, Niccolo Baldassini, and Johannes Wallner. Freeform surfaces from single curved panels. *ACM Trans. Graphics*, 27(3), 2008. Proc. SIGGRAPH.

[21] Helmut Pottmann, Alexander Schiftner, and Johannes Wallner. Geometry of architectural freeform structures. *International Mathematical News (Internationale Mathematische Nachrichten)*, 209:15–28, 2008.

[22] Alla Sheffer, Emil Praun, and Kenneth Rose. Mesh parameterization methods and their applications. *Found. Trends. Comput. Graph. Vis.*, 2(2):105–171, 2006.

[23] Vitaly Surazhsky, Tatiana Surazhsky, Danil Kirsanov, Steven J. Gortler, and Hugues Hoppe. Fast exact and approximate geodesics on meshes. *ACM Trans. Graph.*, 24(3):553–560, 2005.

[24] Petr Vanček and Ivana Kolingerová. Technical section: Comparison of triangle strips algorithms. *Computers and Graphics*, 31(1):100–118, 2007.